12-2013

Towards a System for Controlling Client-Server Traffic in Virtual Worlds using SDN

Jason Anderson
Clemson University, jwa2@clemson.edu

Jim Martin
Clemson University, jmarty@clemson.edu

Towards a System for Controlling Client-Server Traffic in Virtual Worlds using SDN

Jason Anderson
Clemson University
jwa2@clemson.edu

Jim Martin
Clemson University
jmarty@clemson.edu

***Abstract*—Scaling virtual worlds in the age of cloud computing is complicated by the problem of efficiently directing client-server traffic in the face of agile and dynamic compute resources. In this proposed model, Software Defined Networking and compact encoding of avatar data in packet headers are combined to make a fast, scalable, high capacity proxy server that can hide the server infrastructure while fitting well with the IaaS paradigm of modern cloud providers.**

I. Introduction

Scalability is an important factor in the design of virtual worlds. Content is typically designed to fit within the constraints of the computation, memory, storage, and network capacity available to the system. The "pie in the sky" of virtual worlds is an architecture that does not impose limits on content or the number of connected players. The current state of the art architecture in virtual worlds is the distributed client-server model, where the world state can be spread across resources beyond those available on a single compute node.

One well-researched method for partitioning the world state is a dynamic spatial partition mechanism described in [1]. With DSP, the game state is split and migrated to compute resources allocated from a pool as the load increases, and can be merged to reduce the number of partitions as the load decreases. Partitions can also be replicated to other locations for low latency and redundancy. This compartmentalization and replication of the game state to fit compute resources lends itself well to the cloud computing model, where partitions can be created, replicated, destroyed, and migrated to achieve acceptable quality of service and cost effectiveness [2].

A key challenge in a distributed server model is the efficient direction of network traffic. The server responsible for handling a particular client can change at any time, such as during a partition, merge, or migration event, or when the player crosses a partition boundary in the game world. These changes require extra round-trip traffic as the client's traffic destination is renegotiated.

To completely hide the compute resources and game architecture from the clients, Quax *et al* propose a ring of proxy servers that intercept and redirect all traffic between clients and servers [3]. In this model, clients could connect to a nearby proxy to minimize latency. This would act as a more robust form of network address translation (NAT), at the cost of higher overhead due to traffic being passed through the proxy's network stack for application-layer routing decisions.

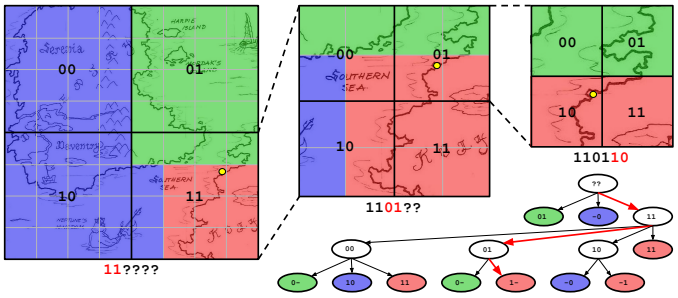

Fig. 1. Recursive quadtree partitioning of a world map where each quadrant is identified by two bits. Logic Servers are assigned sections of the map, indicated by like-colored partitions. In this example, the location described by '110110' can be mapped to the 'red' section by descending the tree.

A promising technology that can address these issues is Software Defined Networking, in which a central controller manages the data plane of multiple physical switches through a protocol such as OpenFlow. The control software installs *flows* on the switch hardware that can match data in the packet headers and perform low level manipulation at high speeds.

The primary contribution of this work is to propose the design of an efficient, scalable SDN-based proxy system that uses information about the player encoded in the transport layer header to make fast routing decisions.

II. Design

A. Architecture

In this model, dynamic quadtree partitioning is utilized to progressively subdivide the virtual world [4], as shown in Fig. 1. One or more partitions are assigned to each Logic Server (LS) for computation of all objects and avatars contained within them. A Proxy Server (PS) intercepts all traffic between a client and LS as proposed in [3]. The PS is implemented as an OpenFlow-enabled switch at the edge of the compute cluster. A Region Manager (RM) node controls region partitioning, merging, migration, and allocation of compute resources as the load demands, and also serves as a controller communicating with the PS via OpenFlow. The PS is assigned a virtual IP address to which clients direct their traffic.

B. Function

The client software determines the location of the player's avatar on the virtual world map using quadtree partitioning as described in [5], up to 10 levels deep. The resulting 20-bit number is written in the Flow Label field of each outgoing IPv6 packet.

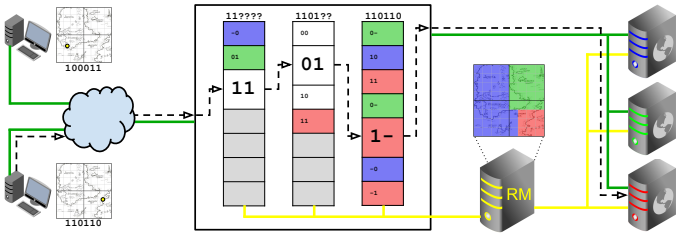

Fig. 2. An SDN-based proxy server implementation. A client encodes the location '110110' in the outgoing packet headers; multiple flows are used to identify the responsible server. The packet's destination address is altered before being sent out on the cluster network. A region management node is responsible for assigning map divisions and updating the proxy's flow tables.

For each LS, the RM installs flows onto the PS that match against data encoded in the IPv6 header. Through pipeline processing [6], flows apply logic equivalent to descending the quadtree: at each step, the switch masks all but two bits of the IPv6 flow label field to match a flow, as shown in Fig. 2. The matching flow either sends the packet to the next flow table for further classification, or acts as a 'leaf' flow which alters the destination IP address, and then directs the packet onto the internal network for normal routing to the LS.

Partition and merge events are handled by adding or removing flows on the PS to be consistent with the quadtree partition scheme. If a region is migrated to a different LS, the leaf flow representing that region is updated with the new IP address. Return traffic from the server to the client and other management traffic can proceed through normal channels.

C. Scaling

The PS would typically perform edge routing for compute clusters acting as regional points of presence for managing latency. It could also be replicated as needed to handle traffic load. Since all flows in the PS would be proactively installed as the partition map changed, network traffic between a central controller and PS instances would be trivial. We expect the client capacity to scale linearly with the number of proxy servers, as long as the LS resources are sufficient.

III. Potential Benefits

Redirecting traffic with a proxy server instead of traditional NAT or direct client-server connections can lead to numerous benefits, including hiding the cluster infrastructure, eliminating connection negotiation traffic on server handoffs, and simple redirection of traffic to state replicas for redundancy and load balancing. While the decision process and traffic redirection described in this work could be performed by a standard middleware server, there are benefits to implementing the logic on an OpenFlow-enabled switch.

First, TCAM-based packet matching and switching on modern OpenFlow-enabled switch hardware is very fast; the pipelined approach to multiple flow matching allows for processing packets at line rate, comparable to the speed of a industry standard switches [7]. To verify this, we built two simple packet redirecting proxy servers: one on a Pronto 3240 48-port SDN switch, the other on a dual-core Opteron machine with 6 Gigabit Ethernet ports. Each proxy was loaded with background full-duplex network traffic while client-server latency was measured, as shown in Fig. 3. We found that the latency cost of the CPU-based implementation increased

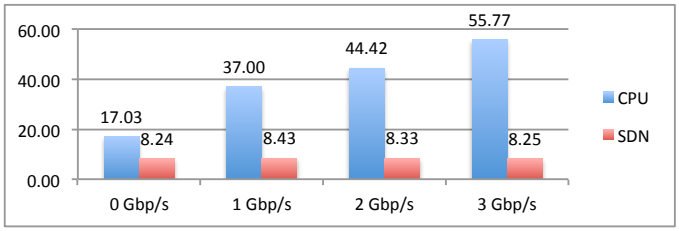

Fig. 3. Latency cost (μs) of a Layer-7 CPU-based proxy compared to flow matching on an SDN switch as background network traffic is increased.

linearly with the background traffic that it handled, while the SDN proxy stayed close to the manufacturer's rating of 8 μsec.

Furthermore, this model is well suited for the movement toward combining SDN and IaaS, where datacenter operators can allow clients to control logical 'slices' of the network to manage their own traffic by running their own OpenFlow controller [8]. Strategic deployment of proxy servers and dynamic provisioning of compute resources could yield a system that scales to meet the demands of millions of clients.

IV. Ongoing Work

Our lab has built a 12-node experimental SDN cluster to conduct research on application-aware networks. In the short term, we will continue to evaluate the performance and capacity of an SDN proxy server compared to more traditional solutions such as NAT and application layer proxies. Our long term aim is to use multiple instances of the SDN proxy server in a large-scale simulation to evaluate its ability to scale as the clients and servers increase.

Acknowledgment

This research is supported in part by NSF Grant #1212680.

References

[1] H. Liu and M. Bowman, "Scale virtual worlds through dynamic load balancing," in *2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2010, pp. 43–52.

[2] D. Maggiorini and L. A. Ripamonti, "Cloud computing to support the evolution of massive multiplayer online games," in *ENTERprise Information Systems*, ser. Communications in Computer and Information Science, M. M. Cruz-Cunha, J. Varajo, P. Powell, and R. Martinho, Eds. Springer Berlin Heidelberg, Jan. 2011, no. 220, pp. 101–110.

[3] P. Quax, J. Dierckx, B. Cornelissen, and W. Lamotte, "ALVIC versus the internet: redesigning a networked virtual environment architecture," *Int. J. Comput. Games Technol.*, vol. 2008, pp. 4:1–4:9, Jan. 2008.

[4] A. Steed and R. Abou-Haidar, "Partitioning crowded virtual environments," in *Proceedings of the ACM symposium on Virtual reality software and technology*, 2003, pp. 7–14.

[5] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *Image Processing, IEEE Transactions on*, vol. 3, no. 3, pp. 327–331, 1994.

[6] Open Networking Foundation, "OpenFlow switch specification 1.1.0," 2011.

[7] O. Ferkouss, I. Snaiki, O. Mounaouar, H. Dahmouni, R. Ben Ali, Y. Lemieux, and C. Omar, "A 100Gig network processor platform for openflow," in *2011 7th International Conference on Network and Service Management (CNSM)*, 2011, pp. 1–4.

[8] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 79–84.